REPAST STATECHARTS GUIDE

JONATHAN OZIK - REPAST DEVELOPMENT TEAM

0. Before we Get Started

Before we can do anything with Repast Simphony, we need to make sure that we have a proper installation of Repast Simphony 2.1. Instructions on downloading and installing Repast Simphony on various platforms can be found on the Repast website.

1. Getting Started with Statecharts

2. States

One of the fundamental building blocks of statecharts are states. Here we introduce the different types of states that existing within the Repast Simphony statecharts framework.

2.1. **Entry State Marker.**

Every statechart must have an entry state marker. This defines the path through which the statechart begins when it is activated.

2.2. **Simple State.**

A simple state looks like Figure 1. At any one point in time within an active statechart, one and only one of the simple states will be active. In addition to their *ID*, simple states can have *On Enter* and *On Exit* actions defined, as seen in the simple state properties panel in Figure 2. These actions are triggered when entering or exiting the simple state, respectively. The keywords available within the action blocks are:

- **agent:** This is the agent that contains the statechart. Any method (e.g., `customMethod`) defined on the agent can be invoked through this reference (e.g., `agent.customMethod()`).
- **state:** This is the state itself. The state's *ID* can be accessed via: `state.getId()`.
- **params:** This is the model's `Parameters` object. As an example, a double valued parameter "dParam" can be retrieved with: `params.getDouble("dParam")`[1].

As is the case with all types of action blocks, the code within them can be defined using Java, Groovy or ReLogo. Any Java, Groovy or ReLogo code can be used to specify the type of behavior that should be executed upon entry to or exit from the state[2].

Date: June 24, 2013.

[1]See the source or JavaDocs for `repast.simphony.parameter.Parameters` for all of the available methods.

[2]When using the ReLogo option, the `agent` parameter is implicit so `customMethod()` is equivalent to `agent.customMethod()`.

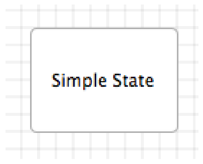
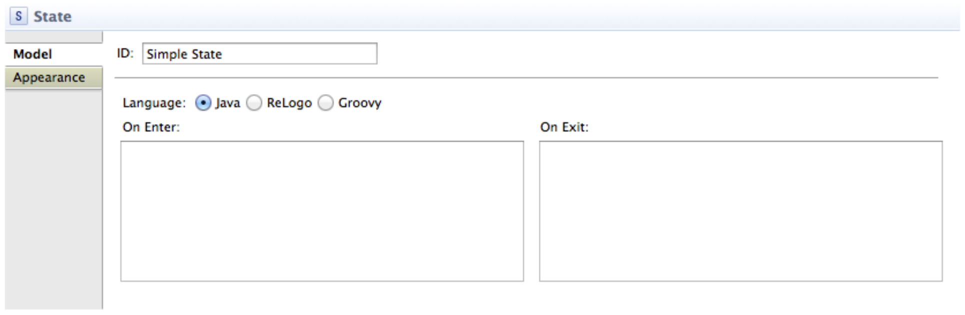


Figure 1. Simple state.



ID: Simple State

Language: [✓ Java] ReLogo Groovy

On Enter:

On Exit:

Figure 2. Simple state properties.

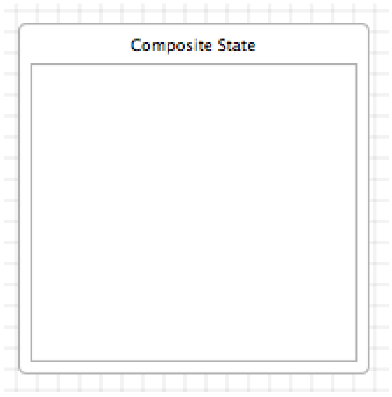
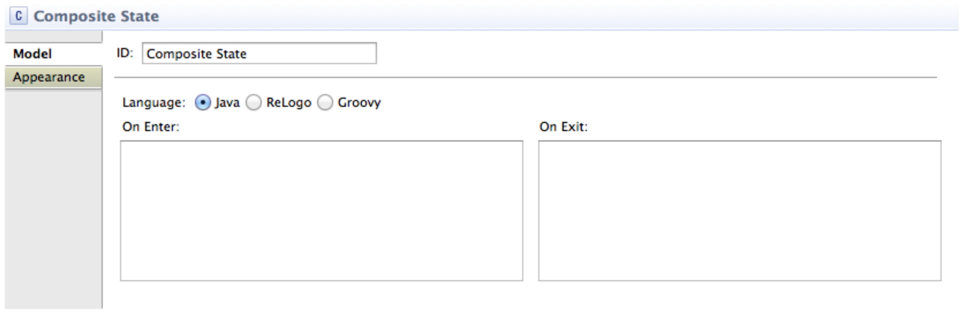


FIGURE 3. Composite state.


Composite State

Model

Appearance

ID: Composite State

Language: [✓ Java] ReLogo Groovy

On Enter:

On Exit:

FIGURE 4. Composite state properties.

2.3. **Composite State.**

Composite states are used to nest elements within a statechart. Figure 3 shows an empty composite state. Composite states can include the following elements:

- Simple state (Section 2.2)
- Composite state (Section 2.3)
- Initial state marker (Section 2.4)
- History state (Section 2.5)
- Final state (Section 2.6)
- Branching state (Section 2.7)

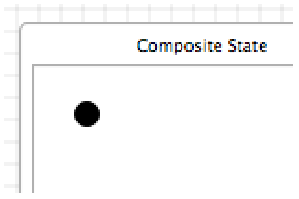


FIGURE 5. Initial state marker (within a composite state).

2.4. **Initial State Marker.** ●

2.5. **History State.** Ⓗ Ⓗ*

2.6. **Final State.** ◉

2.7. **Branching State.** ◇

3. TRANSITIONS

3.1. **Always Transition.**

3.2. **Timed Transition.**

3.3. **Probability Transition.**

3.4. **Condition Transition.**

3.5. **Exponential Decay Rate Transition.**

3.6. **Message Transition.**

3.6.1. *When Message Meets Condition.*

3.6.2. *When Message Equals.*

3.6.3. *When Message is of Class.*

3.6.4. *Always.*